Bounded Unpopularity Matchings*

Chien-Chung Huang[1,**], Telikepalli Kavitha[2,***], Dimitrios Michail[3,†], and Meghana Nasre[2]

[1] Max-Planck-Institut für Informatik, Saarbrücken, Germany. villars@mpi-inf.mpg.de
[2] Indian Institute of Science, India. {kavitha,meghana}@csa.iisc.ernet.in
[3] Department of Informatics and Telematics, Harokopion University of Athens, Greece. michail@hua.gr

Abstract. We investigate the following problem: given a set of jobs and a set of people with preferences over the jobs, what is the optimal way of matching people to jobs? Here we consider the notion of *popularity*. A matching M is popular if there is no matching M' such that more people prefer M' to M than the other way around. Determining whether a given instance admits a popular matching and, if so, finding one, was studied in [3]. If there is no popular matching, a reasonable substitute is a matching whose *unpopularity* is bounded. We consider two measures of unpopularity - *unpopularity factor* denoted by $u(M)$ and *unpopularity margin* denoted by $g(M)$. McCutchen recently showed that computing a matching M with the minimum value of $u(M)$ or $g(M)$ is NP-hard, and that if G does not admit a popular matching, then we have $u(M) \geq 2$ for all matchings M in G.

Here we show that a matching M that achieves $u(M) = 2$ can be computed in $O(m\sqrt{n})$ time (where m is the number of edges in G and n is the number of nodes) provided a certain graph H admits a matching that matches all people. We also describe a sequence of graphs: $H = H_2, H_3, \ldots, H_k$ such that if H_k admits a matching that matches all people, then we can compute in $O(km\sqrt{n})$ time a matching M such that $u(M) \leq k-1$ and $g(M) \leq n(1-\frac{2}{k})$. Simulation results suggest that our algorithm finds a matching with low unpopularity in random instances.

1 Introduction

The problem of assigning people to positions is a very common problem that arises in many domains. The input here is a bipartite graph $G = (\mathcal{A} \cup \mathcal{P}, \mathcal{E})$, where nodes on one side of the bipartite graph rank edges incident on them

* A preliminary version of this work appeared in 11th Scandinavian Workshop on Algorithm Theory (SWAT 08).

** Part of this work was done when the author was visiting Max-Planck-Institut für Informatik, Saarbrücken, Germany.

*** Work done as part of the DST-MPG partner group "Efficient Graph Algorithms" at IISc Bangalore.

† Part of this work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme.

in an order of preference, possibly involving ties. That is, the edge set $\mathcal{E}$ is partitioned into $\mathcal{E}_1 \dot\cup \mathcal{E}_2 \ldots \dot\cup \mathcal{E}_r$. We call $\mathcal{A}$ the set of *applicants*, $\mathcal{P}$ the set of *posts*, and $\mathcal{E}_i$ the set of edges with *rank i*. If $(a,p) \in \mathcal{E}_i$ and $(a,p') \in \mathcal{E}_j$ with $i < j$, we say that *a prefers p* to p'. If $i = j$, then a is *indifferent* between p and p'. The ordering of posts adjacent to a is called a's *preference list*. The problem is to assign applicants to posts that is *optimal* with respect to these preference lists.

This problem has been well-studied in economics literature, see for example [1, 19, 21]. It models some important real-world markets, including the allocation of graduates to training positions [8], families to government-owned housing [20], and mail-based DVD rental systems such as NetFlix. Instances of these markets can be regarded as restricted stable marriage instances [4, 7], in which members of one side of the market (posts) are indifferent between members of the other side of the market (applicants).

A *matching* M of G is a subset of $\mathcal{E}$, such that no two edges of M share a common endpoint. Various criteria have been proposed to measure the "goodness" of a matching. For example, a matching is *Pareto-optimal* [2, 1, 19] if no applicant can improve his/her allocation (say by exchanging posts with another applicant) without requiring some other applicant to be worse off. There are many Pareto-optimal matchings and so we need stronger definitions: a matching is *rank-maximal* [10] if it allocates the maximum number of applicants to their first choice, and then subject to this, the maximum number to their second choice, and so on. Such a matching has the lexicographically maximum *signature* $(n_1, n_2, \ldots)$ where n_i is the number of people assigned to positions they respectively rank i-th. A matching is *maximum utility* if it maximizes $\sum_{(a,p)\in M} u_{a,p}$, where $u_{a,p}$ is the utility of allocating post p to applicant a. Note that $u_{a,p}$ would be a function of the numerical rank that a associates with the edge (a,p). Thus most of these criteria use the actual values or numerical ranks expressed by applicants in their preference lists. As the preference lists only express "relative" ranking of the options, measuring the optimality of a matching as a function of the actual numerical ranks may not be the correct approach. One criterion that does not use numerical ranks is *popularity*. We define it below.

We say that an applicant *a prefers* matching M' to M if (i) a is matched in M' and unmatched in M, or (ii) a is matched in both M' and M, and a prefers $M'(a)$ to $M(a)$ (where $M(a), M'(a)$ are the posts that a is matched to in M and in M', respectively).

Definition 1. *M' is* more popular than *M, denoted by $M' \succ M$, if the number of applicants that prefer M' to M is greater than the number of applicants preferring M to M'. A matching M^* is* popular *if there is no matching M' that is more popular than M^*.*

Popular matchings can be considered *stable* in the sense that an applicant majority vote cannot force a migration to another matching. Also, as mentioned earlier, the notion of popularity does not use numerical ranks expressed by the applicants but only their *relative ranks*. Based on these two properties, popular matchings is one of the optimal ways to assign applicants to jobs bearing their preferences in mind.

However, a popular matching need not always exist, thus popular matchings do not provide a complete answer to the problem of assigning applicants to posts. Figure 1 contains an example instance in which $\mathcal{A} = \{a_1, a_2, a_3\}$, $\mathcal{P} = \{p_1, p_2, p_3\}$, and each applicant prefers p_1 to p_2, and p_2 to p_3. Consider the three symmetrical matchings $M_1 = \{(a_1, p_1), (a_2, p_2), (a_3, p_3)\}$, $M_2 = \{(a_1, p_3), (a_2, p_1), (a_3, p_2)\}$ and $M_3 = \{(a_1, p_2), (a_2, p_3), (a_3, p_1)\}$. None of these matchings is popular, since $M_1 \prec M_2$, $M_2 \prec M_3$, and $M_3 \prec M_1$. In fact, it turns out that this instance admits no popular matching, the problem being that the *more popular than* relation is not transitive.

$$
\begin{array}{lcccc}
a_1 & : & p_1 & p_2 & p_3 \\
a_2 & : & p_1 & p_2 & p_3 \\
a_3 & : & p_1 & p_2 & p_3
\end{array}
$$

Fig. 1. An instance for which there is no popular matching.

Given a bipartite graph $G(\mathcal{A} \cup \mathcal{P}, \mathcal{E})$ with a preference list associated with each applicant, the popular matching problem is to determine if the input instance admits a popular matching, and to find such a matching, if one exists. The first polynomial-time algorithms for this problem were given in [3]: when there are no ties in the preference lists, the problem can be solved in $O(n+m)$ time, where $n = |\mathcal{A} \cup \mathcal{P}|$ and $m = |\mathcal{E}|$, and more generally, the problem can be solved in $O(m\sqrt{n})$ time. The main drawback of the notion of popular matchings is that such matchings may not exist in the given graph. In this situation, it would be desirable if we can find some good substitutes for a popular matching. This motivates our paper.

1.1 Problem Definition

In this paper, we assume that the input instance G does not admit a popular matching. Our goal is to compute a *least unpopular* matching. We use two criteria given by McCutchen [16] to measure the unpopularity of a matching. We first need the following definitions.

Given any two matchings X and Y in G, define $\phi(X,Y) =$ the number of applicants that prefer X to Y. Let us define the following functions to compare two matchings X and Y:

$$\Delta(X,Y) = \begin{cases} \phi(Y,X)/\phi(X,Y) & \text{if } \phi(X,Y) > 0 \\ \infty & \text{otherwise.} \end{cases}$$

and $\delta(X,Y) = \phi(Y,X) - \phi(X,Y)$.

Using the above functions, we can define the *unpopularity factor* of a matching M. Let $\mathcal{M}$ denote the set of all matchings in G and let $\mathcal{Z}$ denote the set of matchings $\tilde{M}$ such that $\phi(M,\tilde{M}) = \phi(\tilde{M},M) = 0$.

$$u(M) = \max_{M' : M' \in \mathcal{M} \text{ and } M' \notin \mathcal{Z}} \Delta(M,M').$$

The *unpopularity margin* of a matching M is defined as:

$$g(M) = \max_{M' : M' \in \mathcal{M}} \delta(M,M').$$

The functions $u(\cdot)$ and $g(\cdot)$ were first introduced by McCutchen, who also gave polynomial time algorithms to compute $u(M)$ and $g(M)$ for any given matching M. A matching M is popular if and only if $u(M) \leq 1$ and $g(M) = 0$. When G does not admit popular matchings, we are interested in computing a matching M with a low value of $u(M)$. Suppose $u(M) \leq 2$. Then such a matching can be considered "reasonably popular" in a model where we say that a matching M' *beats* another matching M only when the number of applicants who prefer M' to M is more than twice the number of applicants who prefer M to M'. If $u(M) \leq 2$, then no other matching can beat M by the above rule. Note that all the 3 matchings M_1, M_2, M_3 described in Figure 1 have their u value equal to 2 and their g value equal to 1. Let us now define a *least unpopular* matching.

Definition 2. *A matching M that achieves the minimum value of $u(M)$ among all the matchings in G is defined as a least unpopularity factor matching in G. Similarly, a matching that achieves the minimum value of $g(M)$ among all matchings in G is defined as a least unpopularity margin matching in G.*

McCutchen recently showed that computing either a least unpopularity factor matching or a least unpopularity margin matching is NP-hard. He also showed that the unpopularity factor of any matching is always an integer. Thus when G does not admit a popular matching, the best matching in terms of the unpopularity factor that one can hope for in G is a matching M that satisfies $u(M) = 2$. Complementing McCutchen's results, we have the following new results here.

- A least unpopularity factor matching can be computed in $O(m\sqrt{n})$ time provided a certain graph H admits an $\mathcal{A}$-*complete* matching. (An $\mathcal{A}$-complete matching means all nodes in $\mathcal{A}$ are matched.) Such a matching M that we compute in H satisfies $u(M) = 2$.
- We also show a more general result. We construct a sequence of graphs: $H = H_2, H_3, \ldots, H_k, \ldots$ and show that if H_k admits an $\mathcal{A}$-complete matching, then we can compute in $O(km\sqrt{n})$ time a matching M such that $u(M) \leq k-1$ and $g(M) \leq n(1-\frac{2}{k})$.
- We ran our algorithm on random graphs using a similar setup as in [3]. In our simulation instances we observe, that H_k admits an $\mathcal{A}$-complete matching for values $k \leq 4$. Thus, in our generated instances our algorithm computes a matching M whose unpopularity factor is a number ≤ 3 and whose unpopularity margin can be upper bounded by $n/2$. We also give a probabilistic analysis to upper bound the performance of our algorithm.
  We continue our simulations with instances which are "highly correlated". Highly correlated instances are those in which applicants have an almost identical preference list. These instances appear more difficult for our new algorithm, which computes matchings with significantly higher unpopularity than in the random case.

1.2 Background and Related Results

The notion of popular matchings was first introduced by Gärdenfors [5] in the context of the stable marriage problem [4]. It is well known that every stable marriage instance admits a weakly stable matching [9] (one for which there is no pair who strictly prefer each other to their partners in the matching). In fact, there can be an exponential number of weakly stable matchings, and so Gärdenfors considered the problem of finding one with additional desirable properties, such as popularity. Gärdenfors showed that when preference lists are strictly ordered, every stable matching is popular. He also showed that when preference lists contain ties, there may be no popular matching.

When only one side has preferences, Abraham et al. [3] gave polynomial time algorithms to find a popular matching, or to report none exists. Recently, Mahdian [14] showed that a popular matching exists with high probability, when (i) preference lists are randomly constructed, and (ii) the number of posts is a factor of $\alpha \approx 1.42$ larger than the number of applicants. He in fact showed a phase transition at α, that is, if the number of posts is smaller than α times the number of applicants, then with high probability popular matchings do not exist.

Manlove and Sng [15] generalized the algorithms of [3] to the case where each post has an associated *capacity*, the number of applicants that it can accommodate. (They described this in the equivalent context of the house allocation

problem.) They gave an $O(\sqrt{C}n_1 + m)$ time algorithm for the no-ties case, and an $O((\sqrt{C}+n_1)m)$ time algorithm when ties are allowed, where n_1 is the number of applicants, m, as usual, is the total length of all preference lists, and C is the total capacity of all of the posts. While the problem of deciding if an input instance with fixed capacities admits a popular matching or not is solvable in polynomial time, Kavitha and Nasre [13] showed that this problem with *variable capacities* is NP-hard. More precisely, they showed that given $G = (\mathcal{A} \cup \mathcal{P}, \mathcal{E})$ that does not admit a popular matching, the problem of deciding if by increasing the capacities of some posts from 1 to 2, the resulting graph admits a popular matching or not is NP-hard.

In [18] Mestre designed an efficient algorithm for the *weighted* popular matching problem, where each applicant is assigned a priority or weight, and the definition of popularity takes into account the priorities of the applicants. In this case his algorithm for the no-ties version has $O(n+m)$ complexity, and for the version that allows ties, the complexity is $O(\min(k\sqrt{n}, n)m)$, where k is the number of distinct weights assigned to applicants.

Assume that there is no tie in the preferences, Kavitha and Nasre [12] designed an $O(n^2+m)$ algorithm for finding a popular matching fulfilling various optimality conditions like fairness, rank-maximality; McDermid and Irving [17] improved the time complexity to $O(n \log n + m)$. They also showed that the problem of computing a minimum weight popular matching can be solved in $O(n+m)$ time.

Though popular matchings need not always exist, Kavitha et al. [11] showed that every instance admits a probability distribution over matchings, also called a *mixed matching*, that is popular. For mixed matchings P and Q, define $\phi(P,Q)$ to be the expected number of applicants that prefer P to Q. A mixed matching P is popular if it satisfies $\phi(P,Q) \geq \phi(Q,P)$ for any mixed matching Q. They also showed a polynomial time algorithm for computing a popular mixed matching.

Organization of the paper. In Section 2 we describe the popular matching algorithm from [3], which is the starting point of our algorithm. We then describe McCutchen's algorithm to compute the unpopularity factor of a given matching. In Section 3 we describe our algorithm and bound its unpopularity factor and unpopularity margin. In Section 4 we report our experimental results. Section 5 presents a probabilistic analysis of our algorithm.

2 Preliminaries

In this section we first review the algorithmic characterization of popular matchings given in [3]. McCutchen introduced the idea of *Posts-Graph* in order to

compute the unpopularity factor of a matching. Since the proof of our main theorem relies on the concept of *Posts-Graph*, we also review McCutchen's algorithm to compute the unpopularity factor of a matching.

For exposition purposes, as was done in [3], we create a unique strictly-least-preferred post $l(a)$ for each applicant a. In this way, we can assume that every applicant is matched, since any unmatched applicant a can be paired with $l(a)$. From now on, matchings are always $\mathcal{A}$-complete. Also, without loss of generality, we assume that preference lists contain no gaps, i.e., if a is incident to an edge of rank i, then a is incident to an edge of rank $i-1$, for all $i > 1$.

Let $H_1 = (\mathcal{A} \cup \mathcal{P}, \mathcal{E}_1)$ be the graph containing only rank-one edges. Then [3, Lemma 3.1] shows that a matching M is popular in G only if $M \cap \mathcal{E}_1$ is a maximum matching of H_1. Maximum matchings have the following important properties, which we use throughout the rest of the paper.

$M \cap \mathcal{E}_1$ defines a partition of $\mathcal{A} \cup \mathcal{P}$ into three disjoint sets: a node $u \in \mathcal{A} \cup \mathcal{P}$ is *even* (resp. *odd*) if there is an even (resp. odd) length alternating path in H_1 (w.r.t. $M \cap \mathcal{E}_1$) from an unmatched node to u. Similarly, a node u is *unreachable* if there is no alternating path from an unmatched node to u. Denote by $\mathcal{N}$, $\mathcal{O}$ and $\mathcal{U}$ the sets of even, odd, and unreachable nodes, respectively. The following lemma gives the Gallai-Edmonds decomposition [6] which is a well known result in matching theory.

Lemma 1 (Gallai-Edmonds Decomposition). *Let $\mathcal{N}$, $\mathcal{O}$ and $\mathcal{U}$ be the sets of nodes defined by H_1 and $M \cap \mathcal{E}_1$ above. Then*

(a) $\mathcal{N}$, $\mathcal{O}$ and $\mathcal{U}$ are pairwise disjoint, and independent of the maximum matching $M \cap \mathcal{E}_1$.
(b) In any maximum matching of H_1, every node in $\mathcal{O}$ is matched with a node in $\mathcal{N}$, and every node in $\mathcal{U}$ is matched with another node in $\mathcal{U}$. The size of a maximum matching is $|\mathcal{O}| + |\mathcal{U}|/2$.
(c) No maximum matching of H_1 contains an edge between a node in $\mathcal{O}$ and a node in $\mathcal{O} \cup \mathcal{U}$. Also, H_1 contains no edge between a node in $\mathcal{N}$ and a node in $\mathcal{N} \cup \mathcal{U}$.

Using this node partition, we make the following definitions: for each applicant a, $f(a)$ is the set odd/unreachable posts amongst a's most-preferred posts. Also, $s(a)$ is the set of a's most-preferred posts amongst all even posts. We refer to posts in $\cup_{a \in \mathcal{A}} f(a)$ as *f-posts* and posts in $\cup_{a \in \mathcal{A}} s(a)$ as *s-posts*. We note that this definition of $f(a)$ is different from the original definition as in [3]. By our definition, the set $f(a)$ can be empty for some applicant a but $s(a) \neq \emptyset$ for any a, since $l(a)$ is always even. Also note that the set of f-posts and s-posts are disjoint. We remark that there may be posts in $\mathcal{P}$ that are neither f-posts nor s-posts. The next theorem characterizes the set of all popular matchings.

Theorem 1 ([3]). *A matching M is popular in G iff (i) $M \cap \mathcal{E}_1$ is a maximum matching of $H_1 = (\mathcal{A} \cup \mathcal{P}, \mathcal{E}_1)$, and (ii) for each applicant a, $M(a) \in f(a) \cup s(a)$.*

Figure 2 contains the algorithm from [3], based on Theorem 1, for solving the popular matching problem.

Popular-Matching$(G = (\mathcal{A} \cup \mathcal{P}, \mathcal{E}))$
Construct the graph $G' = (\mathcal{A} \cup \mathcal{P}, \mathcal{E}')$, where $\mathcal{E}' = \{(a,p) : a \in \mathcal{A} \text{ and } p \in f(a) \cup s(a)\}$.
Construct a maximum matching M of $H_1 = (\mathcal{A} \cup \mathcal{P}, \mathcal{E}_1)$.
//Note that M is also a matching in G'.
Remove any edge in G' between a node in O and a node in $O \cup \mathcal{U}$.
//No maximum matching of H_1 contains such an edge.
Augment M in G' until it is a maximum matching of G'.
Return M if it is $\mathcal{A}$-complete, otherwise return *"no popular matching"*.

Fig. 2. An $O(\sqrt{n}m)$-time algorithm for the popular matching problem (from [3]).

2.1 McCutchen's algorithm

Here we outline the algorithm given by McCutchen for computing the unpopularity factor of a matching. Given a matching M, the idea is to find a series of promotions (of applicants) at the cost of demoting one applicant. The longest such promotion path determines the unpopularity factor of the particular matching. Such a path can be discovered by building a directed weighted graph on the set of posts, called the *Posts-Graph* $G_{\mathcal{P}}$. The vertices of $G_{\mathcal{P}}$ represent all the posts $\mathcal{P}$ in the original graph. We add edges into $G_{\mathcal{P}}$ based on the following rules: (let $M(p)$ denote the applicant to which post p is matched to in the matching M)

- an edge with weight -1 is directed from post p_i to p_j if $M(p_i)$ prefers p_j to p_i.
- an edge with weight 0 is directed from post p_i to p_j if $M(p_i)$ is indifferent between p_i and p_j.

Note that there is no edge from p_i to p_j if $M(p_i)$ prefers p_i to p_j. The series of promotions mentioned above is a negative weight path in this graph. To find the longest negative weight path in this graph, we add a dummy vertex s with 0 weight edges from s to all posts. An algorithm which finds shortest paths from source s to all posts will give the longest negative weight path in $G_{\mathcal{P}}$. Existence of a negative weight cycle implies that there exists a promotion sequence without any demotion and hence the unpopularity factor of the matching is ∞. Let

us assume that no negative weight cycles exist. Then all posts have a 0 or finite length negative weight shortest path from the source. The post whose distance from the source is the "most negative" determines the unpopularity factor of the matching M. For details of the proof of correctness, refer to [16].

3 Our algorithm

In this section we describe a *greedy* strategy to compute a matching of G, whose unpopularity factor and margin can be bounded. Our algorithm is iterative and in every iteration it constructs a graph H_i and a maximum matching M_i in H_i. We show that if M_i is an $\mathcal{A}$-complete matching, then $u(M_i) \leq i-1$ and $g(M_i) \leq n(1-2/i)$.

We will first give some intuition before we formally describe our algorithm. Recall that the popular matching algorithm first finds a maximum cardinality matching M_1 in the graph H_1 (whose edge set is the set of all rank 1 edges). The algorithm then identifies all even applicants/posts using the Gallai-Edmonds decomposition and adds the edges (a,p) where a is even and $p \in s(a)$ to the pruned graph H_1 (all rank 1 edges between an odd node in H_1 and a node that is odd or unreachable in H_1 are removed from H_1). Note that each such edge (a,p) is *new* to H_1, that is, such an edge is not already present in H_1 since by Gallai-Edmonds decomposition (part (iii)), there is no edge between two *even* vertices of H_1, and here both a and p are even in H_1. In this new graph, call it H_2, M_1 is augmented to a maximum cardinality matching M_2. In case M_2 is $\mathcal{A}$-complete, we declare that the instance admits a popular matching. Otherwise no popular matching exists.

The idea of our algorithm here is an extension of the same strategy. Since we are considering instances that do not admit a popular matching, M_2 found above will not be $\mathcal{A}$-complete. In this case, we go further and find the Gallai-Edmonds decomposition of nodes in H_2 and identify nodes that are even in H_1 and in H_2. A node that is odd or unreachable in either H_1 or in H_2 will always be matched by a maximum cardinality matching in H_2 that is obtained by augmenting a maximum cardinality matching in H_1. Hence the nodes that are not guaranteed to be matched by such a matching M_2 are the applicants and posts that are even in both H_1 and H_2.

So let us now add the edges (a,p) to H_2 where a and p are nodes that are even in both H_1 and H_2 and among all posts that are even in both H_1 and H_2, p is a most preferred post of a. We would again like to point out that such an edge (a,p) did not exist in either H_1 or in H_2, since a and p were even in H_1 and in H_2. We also prune H_2 to remove edges that are contained in no maximum cardinality matching of H_2 and call the resulting graph H_3. We then

augment M_2 to get M_3 and continue the same procedure till we finally get an $\mathcal{A}$-complete matching M_i. Note that, for any applicant a, its unique last-resort post $l(a)$ remains even till the edge $(a, l(a))$ gets added to our graph. Further, once the edge $(a, l(a))$ is added, applicant a becomes odd or unreachable and hence gets marked. This ensures that our iterative algorithm is guaranteed to halt and output an $\mathcal{A}$-complete matching.

We would now like to contrast our approach above with the approach used in the algorithm for *rank-maximal* matchings [10]. In the i-th iteration the algorithm for rank-maximal matchings would add edges from an applicant a that is even in each of the previous iterations to a post p that was even in each of the previous iterations only if p was a rank i post in a's preference list. On the other hand, our algorithm will add an edge from an applicant a that is even in each of the previous iterations to a post q that is even in each of the previous iterations if q is a's most preferred post among all such posts. Note that the rank of the edge (a, q) is not necessarily i. Thus the absolute ranks in the preference lists are not important and instead, what is important here is the relative ordering of posts in each applicant's preference list. Thus unlike in the rank-maximal matching algorithm, in our algorithm every applicant a that has been even in all previous iterations will have some new edge incident on it in the i-th iteration.

We give an example below to contrast rank-maximal matchings with the matching computed by our algorithm. Our instance consists of n applicants $\{a_1, a_2, \cdots, a_{n/2}, b_1, b_2, \cdots, b_{n/2}\}$. The preference lists of applicants $a_1, \cdots, a_{n/2}$ are as shown in Figure 3. The remaining $n/2$ applicants, $b_1, \ldots, b_{n/2}$ have preference lists of length 1 each. The preference list of b_i consists of post q_i, for $1 \leq i \leq n/2$. Note that we did not add the last resort posts here as they play no role in this example. This instance admits a popular matching (a matching with unpopularity factor = 1), however the unpopularity factor of the rank-maximal matching in this instance is $n/2 - 1$, as described below.

$$
\begin{array}{rl}
a_1 & : p_1\ p_2 \\
a_2 & : p_1\ p_2\ p_3 \\
a_3 & : p_1\ q_1\ p_3\ p_4 \\
a_4 & : p_1\ q_1\ q_2\ p_4\ p_5 \\
a_5 & : p_1\ q_1\ q_2\ q_3\ p_5\ p_6 \\
a_6 & : p_1\ q_1\ q_2\ q_3\ q_4\ p_6\ p_7 \\
 & \vdots \\
 & \vdots \\
a_{n/2-1} & : p_1\ q_1\ \cdots\ q_{n/2-3}\ p_{n/2-1}\ p_{n/2} \\
a_{n/2} & : p_1\ q_1\ \cdots\ q_{n/2-1}\ p_{n/2}
\end{array}
$$

Fig. 3. The preference lists of the first $n/2$ applicants in our instance.

As popular matchings and rank-maximal matchings are maximum-cardinality matchings on rank 1 edges, these matchings match each b_i to q_i. Note that $f(a_i) = p_1$ for all i, while $s(a_i) = p_i$ for $2 \leq i \leq n/2$. Thus the matching that matches each a_i to p_i for each $1 \leq i \leq n/2$ is popular.

A rank-maximal matching, on the other hand, would match $a_{n/2}$ to p_1 to avoid the edge $(a_{n/2}, p_{n/2})$ of rank $n/2+1$ and match each a_i to p_{i+1} for $1 \leq i \leq n/2-1$. The signature of this matching is $(n/2+1, 1, \ldots, 1, 1)$, which is lexicographically better than the signature of the popular matching, which is $(n/2+1, 1, \ldots, 1, 0, 1)$. However the rank-maximal matching has a promotion path $\langle p_{n/2}, p_{n/2-1}, \ldots, p_1 \rangle$ of length $n/2-1$ by promoting a_i from p_{i+1} to p_i, for $1 \leq i \leq n/2-1$, and demoting $a_{n/2}$ to $p_{n/2}$. This makes the $n/2-1$ applicants $a_1, \ldots, a_{n/2-1}$ better off while only $a_{n/2}$ is worse off. Thus even in instances that admit popular matchings, a rank-maximal matching can have an unpopularity factor as large as $n/2-1$.

3.1 The algorithm

We present our algorithm (which we call Bounded-Unpop in the rest of the paper) here. Our algorithm starts with all nodes (applicants and posts) unmarked and ends when every applicant is marked, that is, when an $\mathcal{A}$-complete matching is found. Figure 4 describes our algorithm.

Initialize $i = 0$ and let all nodes be unmarked.
Let $H_0 = (\mathcal{A} \cup \mathcal{P}, \emptyset)$ and $M_0 = \emptyset$.
While M_i is not $\mathcal{A}$-complete do:

1. Add edges (a, p) to H_i where (i) a is unmarked, (ii) p is unmarked and (iii) p is a's most preferred post among all unmarked posts. Call the resulting graph H_{i+1}.
2. Augment M_i in H_{i+1} to get a new matching M_{i+1} which is a maximum cardinality matching of H_{i+1}.
3. $i = i + 1$.
4. Partition the nodes of $\mathcal{A} \cup \mathcal{P}$ into three disjoint sets: $\mathcal{N}_i, \mathcal{O}_i, \mathcal{U}_i$.
   – $\mathcal{N}_i$ and $\mathcal{O}_i$ consists of nodes that can be reached in H_i from an unmatched node by an even/odd length alternating path with respect to M_i, respectively.
   – $\mathcal{U}_i$ consists of nodes that are unreachable by an alternating path from any unmatched node in H_i.
5. Mark all unmarked nodes in $\mathcal{O}_i \cup \mathcal{U}_i$.
6. Delete all edges of H_i between a node in $\mathcal{O}_i$ and a node in $\mathcal{O}_i \cup \mathcal{U}_i$

Return M_i.

Fig. 4. Algorithm Bounded-Unpop : An $O(km\sqrt{n})$-time algorithm for finding an $\mathcal{A}$–complete matching.

Note that once a post becomes odd or unreachable in any iteration, it gets marked and such a post cannot get any new edges incident upon it in the subsequent iterations. We use this to show that if we find an $\mathcal{A}$-complete matching in the graph H_k, then the unpopularity factor of this matching is bounded by $k-1$. The running time of our algorithm is determined by the least k such that H_k admits an $\mathcal{A}$-complete matching. Since each iteration of our algorithm takes $O(m\sqrt{n})$ time, the overall running time is $O(km\sqrt{n})$, where k is the least number such that H_k admits an $\mathcal{A}$-complete matching.

Before we prove our main theorems, we need the following definition. Recall that $G_{\mathcal{P}}$ is the *Posts-Graph* described in Section 2.1. Let us partition the vertices of $G_{\mathcal{P}}$ into k layers (corresponding to the k iterations): a post belongs to a layer t if it gets marked for the first time in iteration t. Also, let posts that remain even throughout the algorithm (including the k-th iteration) belong to layer k. Let $\ell(p)$ denote the layer number of post p in $G_{\mathcal{P}}$. Lemma 2 shows a very useful property that the above partitioning of vertices of $G_{\mathcal{P}}$ achieves. Recall that if (p,q) is an edge in $G_{\mathcal{P}}$, then $M_k(p)$ likes q at least as much as p.

Lemma 2. *There is no edge in $G_{\mathcal{P}}$ from a lower layer to a higher layer. More precisely,*

(1) if (p,q) is an edge such that $M_k(p)$ strictly prefers q to p, then $\ell(p) > \ell(q)$;
(2) if (p,p') is an edge such that $M_k(p)$ is indifferent between p and p', then $\ell(p) \geq \ell(p')$.

Proof. We first show claim (1). Let (p,q) be an edge in $G_{\mathcal{P}}$ such that $M_k(p)$ strictly prefers q to p. Let p be a post that belongs to layer i. We now show that q should belong to a layer j such that $j < i$.

Note that since p got marked in the i-th iteration, no new edges are ever added to p in any of the subsequent iterations. Further, since the edge $(M_k(p),p)$ exists in the graph H_k, we can conclude that in some iteration $t \leq i$, post p was the most preferred *unmarked* post for $M_k(p)$. Hence all the posts that $M_k(p)$ strictly prefers to p were already marked before the t-th iteration. That is, these posts belong to layers j such that $j < t \leq i$. Thus if (p,q) is an edge such that $M_k(p)$ strictly prefers q to p, then q belongs to layer j, where $j < i$. This proves claim (1).

We now prove claim (2). Let (p,p') be an edge such that $M_k(p)$ is indifferent between p and p' and assume for the sake of contradiction that p belongs to layer i and p' belongs to layer j where $j > i$. Since p gets marked in the i-th iteration, the edge $(M_k(p),p)$ must have been added in some iteration t such that $t \leq i$. The posts p and p' have the same rank in $M_k(p)$'s preference list. Thus in iteration t, when the edge $(M_k(p),p)$ was added to create H_t, the edge $(M_k(p),p')$ also

was added to H_t. If not, it implies that the post p' was already marked and we are done.

Let us consider iteration i. The vertex p gets marked in the i-th iteration, so $p \in O_i \cup \mathcal{U}_i$. Since we delete all edges between vertices in $O_i \cup \mathcal{U}_i$ and vertices in O_i and as the edge $(M_k(p), p)$ does not get deleted from H_i (since this edge is used by M_k), it follows that $M_k(p) \in \mathcal{N}_i \cup \mathcal{U}_i$.

Note that the edge $(M_k(p), p')$ was present in H_t and it has to remain in H_i as p' remains an *even* vertex till the j-th iteration where $j > i$, thus no edge incident to p' in our graph gets removed in any earlier iteration. This gives the required contradiction as there cannot be an edge in H_i between $M_k(p) \in \mathcal{N}_i \cup \mathcal{U}_i$, and $p' \in \mathcal{N}_i$ - such an edge contradicts Gallai-Edmonds decomposition part (iii). Thus p' has to get marked in either iteration i or in an earlier iteration. This proves claim (2). □

The proof of the main theorem, stated below, now follows easily from the above lemma.

Theorem 2. *If our algorithm finds an $\mathcal{A}$-complete matching M_k in H_k, then $u(M_k) \leq k-1$.*

Proof. Recall that the unpopularity factor of M_k is the "most negative" distance of a vertex (post) in $G_{\mathcal{P}}$ from the dummy source s as described in Section 2 when we assign a weight of -1 to each edge (p,q) where $M_k(p)$ strictly prefers q to p and a weight of 0 to each edge (p,p') where $M_k(p)$ is indifferent between p and p'. All the edges (s,p) have weight 0.

Note that any path in $G_{\mathcal{P}}$ between 2 posts can never go from a lower numbered layer to a higher numbered layer (by Lemma 2) as negative weight edges always go from a higher numbered layer to a lower numbered layer and 0 weight edges go either within the same layer or to a lower layer. As there are only k layers, there can be at most $k-1$ negative weight edges in any path between 2 posts. Thus it follows that the most negative distance in $G_{\mathcal{P}}$ from s cannot be less than $-(k-1)$, in other words, $u(M_k) \leq k-1$. □

Theorem 3. *If our algorithm finds an $\mathcal{A}$-complete matching M_k in H_k, then $g(M_k) \leq n(1-\frac{2}{k})$.*

Proof. Let M_k be the $\mathcal{A}$-complete matching produced by our algorithm after k iterations and let M be any other $\mathcal{A}$-complete matching in G. Now let us construct a weighted directed graph $H_{\mathcal{P}}$ similar to the posts graph $G_{\mathcal{P}}$. The vertices of $H_{\mathcal{P}}$ are all posts p such that $M_k(p) \neq M(p)$. For every applicant a we have a directed edge from $M_k(a)$ to $M(a)$ with a weight of $-1, 0, +1$ if a considers $M(a)$ better than, the same as or worse than $M_k(a)$, respectively. Any post p that

does not belong to $H_{\mathcal{P}}$ is matched to the same applicant in M_k as well as in M and hence the corresponding applicant does not contribute to the unpopularity margin. Furthermore, it is clear that the sum of weights of all edges in $H_{\mathcal{P}}$ gives the negative of the unpopularity margin by which M dominates M_k.

First note that $H_{\mathcal{P}}$ is a set of disjoint paths and cycles. This is because, $H_{\mathcal{P}}$ can equivalently be constructed from $S = M_k \oplus M$ by striking off applicants and giving appropriate directions and weights to edges. Thus a path in S continues to be a path in $H_{\mathcal{P}}$ although it may no longer be of even length. The same is true for cycles also. If a path or cycle consists of only 0 weight edges, then we can drop such a cycle/path from the graph, since these edges do not contribute to the unpopularity margin. In addition, note that any cycle or path cannot be composed of only negative and zero weight edges, otherwise the unpopularity factor of M_k is ∞, a contradiction to Theorem 2. Hence we can assume that every cycle or path contains at least one positive edge.

Let ρ be any path or cycle in $H_{\mathcal{P}}$. Furthermore, let α and β be the numbers of -1's and $+1$'s in ρ respectively. We define the function:

$$\text{frac-margin}(\rho) = \frac{\alpha - \beta}{\text{number of edges in } \rho}$$

Let us try to bound frac-margin(ρ) for each ρ. For the sake of simplicity, let us first assume that the preference lists are strict. So there are only ± 1 weight edges in $H_{\mathcal{P}}$. Thus frac-margin$(\rho) = (\alpha - \beta)/(\alpha + \beta)$. Since the unpopularity factor of M_k is bounded by $k-1$, it is easy to see that the unpopularity factor of ρ is also bounded by $k-1$ (refer to [16] for a proof), implying $\alpha/\beta \leq k-1$. Thus $\beta/(\alpha+\beta) \geq 1/k$, and $\alpha/(\alpha+\beta) \leq 1 - 1/k$. Hence frac-margin($\rho$) for any path or cycle ρ is at most $1 - 2/k$. The contribution of ρ towards $\delta(M_k, M)$ is (number of edges in ρ)$\cdot$(frac-margin(ρ)). This is at most $n_\rho(1 - 2/k)$ where n_ρ is the number of edges in ρ. Since a unique applicant a is associated with each edge $(M_k(a), M(a))$ of $H_{\mathcal{P}}$, it follows that $\sum n_\rho \leq n$. Thus $\delta(M_k, M) \leq n(1-2/k)$ where M is any matching.

The proof for the case with ties also follows from the above argument. Since 0 weight edges in ρ do not affect the numerator of frac-margin(ρ) and only increase the denominator of frac-margin(ρ), it is easy to see that frac-margin(ρ) for a path or cycle ρ with 0 weight edges is dominated by frac-margin(ρ') where ρ' is obtained from ρ by contracting 0 weight edges. Thus frac-margin$(\rho) \leq 1 - 2/k$ and thus $\delta(M_k, M) \leq n(1 - 2/k)$ where M is any matching. Thus $max_{M'} \delta(M_k, M) \leq n(1-2/k)$. □

Corollary 1. *Let G be a graph that does not admit a popular matching. If our algorithm produces an $\mathcal{A}$-complete matching M in H_3, then M is a least unpopularity factor matching in G.*

Proof. It follows from Theorem 2 that if our algorithm produces an applicant complete matching M in H_3, then $u(M) \leq 2$. McCutchen [16] showed that the unpopularity factor of any matching is always an integer. Thus if G admits no popular matching, then the lowest value of $u(\cdot)$ we can hope for is 2. Since $u(M) \leq 2$, it follows that this is a least unpopularity factor matching. □

4 Experimental Results

4.1 Random Instances

In this section we present simulation results. For the generated random instances our algorithm is able to find a matching with small unpopularity.

We follow the setting used in [3] so that our experimental results are comparable to those reported in [3]. The number of applicants and posts are equal (denoted by n) and preference lists have the same length l. Existence of ties is characterized by a single parameter t which denotes the probability of an entry in the preference list to be tied with its predecessor.

Table 1 contains simulation results for random graphs with $n = 100$ and $n = 500$ for different values of parameters l and t. The table shows the number of instances (out of 1000 instances) that finish in some particular round of the execution. Round 2 means that the instance has a popular matching. It is easy to observe that the difficult cases are the ones which are denser (l is large) and where we only have a few ties (t is small). For a fixed value of l as t decreases the algorithm requires more rounds until it returns a solution. Note that intuitively ties make the task of finding a popular matching easier. More precisely, if we resolve the ties in all preference lists in an arbitrary way, we obtain an instance without any ties, and every popular matching for this instance is also a popular matching for the original instance.

We study the situation, where l is large and t small, further by varying the value of n in order to see whether our observations for $n = 100$ and $n = 500$ are valid for larger values of n. Let us remark that Mahdian [14] proved the following result. If the right side of the bipartite partition is slightly larger than (≈ 1.42 times) the left side, then the instance has a popular matching with high probability. Since we try to identify difficult instances we keep the two sides of the partition equal, which is also the case in many practical situations, where there are no surplus posts when compared to the number of applicants.

Table 1 shows the number of rounds (again out of 1000) that is required for different values of n when $t = 0.05$ and $l = n$, i.e., the graph is complete bipartite and only a few ties of very small length exist. The table suggests that in our generated instances as n increases the probability of terminating at the smaller rounds decrease and the one of higher rounds increase. However, this

$n = 100$				
		# rounds		
l	t	2	3	4
10	0		1000	
	0.05	4	996	
	0.2	28	972	
	0.5	471	529	
	0.8	729	271	
	1.0	1000		
25	0		988	12
	0.05		991	9
	0.2	3	991	6
	0.5	138	861	1
	0.8	773	227	
	1.0	1000		
50	0		950	50
	0.05		948	52
	0.2	1	978	21
	0.5	158	832	10
	0.8	793	207	
	1.0	1000		
100	0		943	57
	0.05		952	48
	0.2	2	973	25
	0.5	148	836	16
	0.8	783	217	
	1.0	1000		

$n = 500$				
		# rounds		
l	t	2	3	4
10	0		1000	
	0.05		1000	
	0.2		1000	
	0.5	176	824	
	0.8	62	938	
	1.0	1000		
25	0		1000	
	0.05		1000	
	0.2		1000	
	0.5		999	1
	0.8	93	907	
	1.0	1000		
50	0		951	49
	0.05		967	33
	0.2		994	6
	0.5		997	3
	0.8	104	896	
	1.0	1000		
100	0		758	242
	0.05		828	172
	0.2		942	58
	0.5		989	11
	0.8	93	907	
	1.0	1000		

n	# rounds		
	2	3	4
10	585	413	2
25	141	844	15
50	6	962	32
100		952	48
250		896	104
500		820	180
1000		667	333
1500		541	459
2000		320	680

Table 1. The left and middle tables show the number of instances with $n = 100$ and 500 nodes respectively (out of 1000 instances) that finish in round number 2 (popular matching), 3 or 4 for different values of the parameters l and t. The table on the right shows the number of instances (out of 1000 instances) that finish in round number 2 (popular matching), 3 or 4 for fixed $t = 0.05$, $l = n$ and different values of the parameter n. We note that in all tables the sum of the columns in each row sum up to 1000.

is not accompanied with any increase in rounds larger than 4. Due to memory constraints we could not continue the experiment for larger values of n. We will see, however, in the next section instances where rounds higher than 4 are possible.

Our experimental results on random instances are very promising. The algorithm behaves nicely, far away from a possible large approximation.

4.2 Highly Correlated Instances

We continue our simulation results with some randomly structured instances, which are very likely to be the most difficult instances. Recall from Figure 1

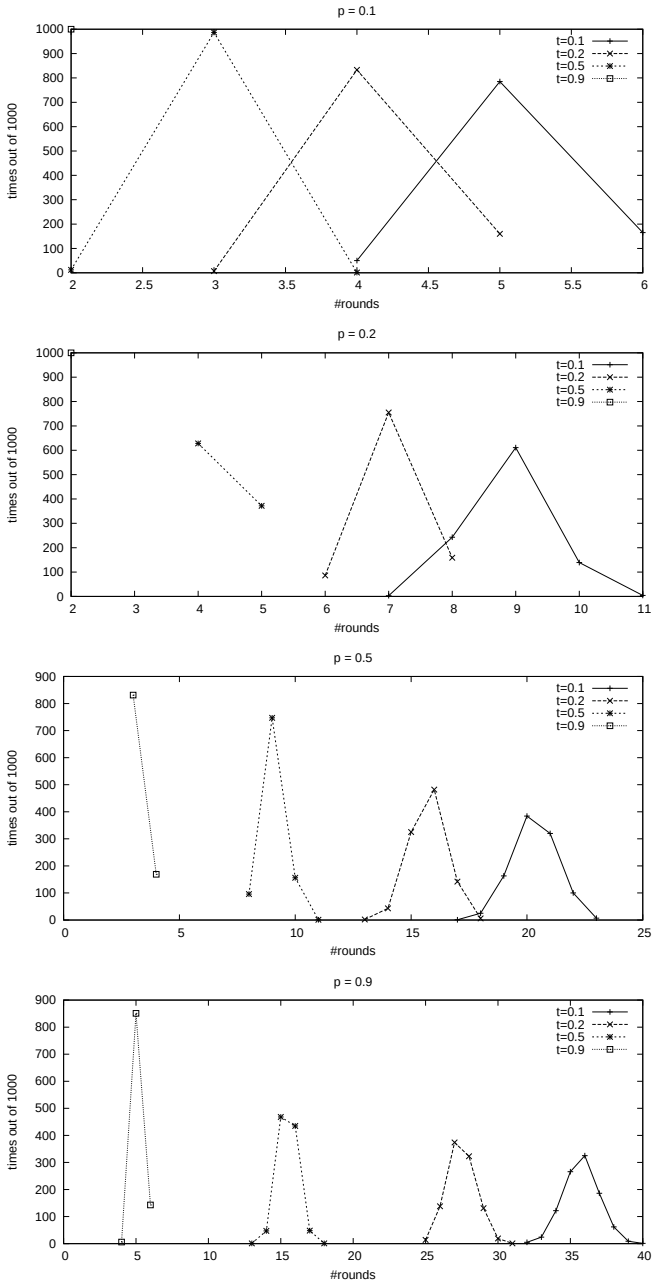

Fig. 5. Distribution of number of rounds needed by our algorithm out of 1000 highly correlated random instances. All have been constructed with $n = 100$. p is the density parameter, t the probability of ties. The distribution of rounds is normally distributed around a mean value, which increases as the tie probability decreases.

that the instance where each applicant ranks all posts in the same way, has no popular matching. Since most of the time applicants rank posts based on the posts' reputation, such instances are very likely to appear in practice.

The setting here is the following. The number of applicants and posts are equal, denoted by n. We again choose to have the same number of applicants and posts due to [14]. Moreover, there is a well known and strict total order R of posts that each applicant is aware of. Every applicant chooses uniformly at random a subset of $n \cdot p$ posts where $0 \leq p \leq 1$ is a density parameter and initially ranks these $n \cdot p$ posts based on the order R.

After applicants select posts, ties are introduced based on the parameter $0 \leq t \leq 1$. Ties are introduced as in the previous setting. Each choice $p \in P$ of an applicant $a \in A$ (except the first) will be a tie with its predecessor with probability t.

In this setting, the instance of Figure 1 is constructed with $n = 3$, order $R = \{p_1, p_2, p_3\}$, $p = 1.0$ and $t = 0.0$. The highly correlated instances are considerable more difficult than the random ones, a fact that is supported by our experiments.

In order to understand the algorithm's behavior we fix $n = 100$ and run our algorithm for different values of the density of the instance p and the probability of ties t. For each pair of values of p and t we run 1000 instances. Figure 5 plots the distribution of rounds taken by our algorithm when executed 1000 times. As can be seen from the figure, with a high tie probability the algorithm terminates in a short number of rounds. When ties are less common, rounds tend to increase. The worst case appears in dense instances with very small tie probability (just like Figure 1).

It is worth noting that in all cases the number of rounds is normally distributed around a mean value which shifts depending on the instance parameters.

4.3 Comparison with Rank-Maximal matchings

For the sake of completeness, we compare experimentally our new algorithm with rank-maximal matchings for the instances of Section 4.1 and Section 4.2. Here we are interested in comparing unpopularity factors. Let us note, however, that in a given instance different rank-maximal matchings might have different unpopularity factors. Thus, we explicitly compare matchings computed by Algorithm Bounded-Unpop and the rank-maximal matchings algorithm presented in [10].

Comparing these two algorithms, we observed that the algorithm in [10] computes matchings with a larger unpopularity factor for all instances that we

random instances							
$n = 100, l = 100, t = 0.05$							
unpopularity factor	2	3	4	5	6	7	8
Bounded-Unpop algorithm	959	41					
Rank-Maximal algorithm in [10]	26	488	407	74	5		
$n = 500, l = 500, t = 0.05$							
unpopularity factor	2	3	4	5	6	7	8
Bounded-Unpop algorithm	833	167					
Rank-Maximal algorithm in [10]			177	552	243	26	2

Table 2. Comparison of the unpopularity factor between the matching computed by our new algorithm and the matchings computed by the rank-maximal matching algorithm in [10]. We only consider here the most difficult random instances (dense instances with very few ties). The table shows the distribution of the unpopularity factor out of 1000 random instances, thus the values of each row sum up to 1000. For example for $n = 100, l = 100, t = 0.05$ the rank-maximal matching algorithm in [10] computes a matching with unpopularity factor 3 in 488 instances out of a 1000.

run. Therefore, we present the results only for the most difficult instances. Table 2 contains the distribution (out of 1000 instances) of the unpopularity factor in random instances that are dense and with very few ties.

Highly correlated instances demonstrate the same behavior. The only difference is that the unpopularity factor shifts to the right. For $n = 100$, $p = 0.9$ and $t = 0.1$ the new algorithm computes matchings with unpopularity factor ranging between 31 and 39. At the same time the rank-maximal matchings algorithm in [10] computes matchings with unpopularity factor between 42 and 56. Similarly for $n = 500$, $p = 0.9$ and $t = 0.1$ the new algorithm computes matchings with unpopularity ranging from 129 and 140 while the algorithm in [10] from 221 to 251.

5 A bound on the number of iterations taken by our algorithm

In this section we give an upper bound on the expected number of iterations our algorithm takes to compute an $\mathcal{A}$-complete matching on *random* instances. In this section, we assume that each preference list is complete and has no ties in it. Each preference list is a uniform random permutation on the set of all posts. Let n_0 denote both the number of applicants and the number of posts. We show that the expected number of iterations taken by our algorithm on such an instance is at most $\ln n_0 + 1$.

We now describe our random experiment. Each applicant picks a permutation independently and uniformly at random from the set of all permutations on the posts. For the sake of analysis, we view the experiment in a slightly different

manner as was done in [14]. Each applicant picks his/her first choice post independently and uniformly at random from the set of posts $\mathcal{P} = \{p_1, \ldots, p_{n_0}\}$. Let M_1 denote a maximum cardinality matching in the graph where each applicant adds edges to his/her first choice post as picked above. We use Gallai-Edmonds decomposition to partition the nodes into $O_1, \mathcal{U}_1$, and $\mathcal{N}_1$. Note that every post in $O_1 \cup \mathcal{U}_1$ has one or more than one applicant adjacent to it. We match each of these posts to one of its adjacent applicants as follows: every post $p \in \mathcal{U}_1$ has a unique applicant a adjacent to it, so we match p to a; every post $p' \in O_1$ has more than one applicant adjacent to it, we choose an a' arbitrarily from these applicants and match p' to a'. For any $p_j \in \mathcal{P}$, the post p_j is unmatched if no applicant chose p_j. Thus it is analogous to a balls and bins experiment where we drop each ball uniformly at random into one of the bins. Hence we have:

$$Pr(p_j \text{ is unmatched after the first iteration}) = \left(1 - \frac{1}{n_0}\right)^{n_0} \leq \frac{1}{e}.$$

Let n_1 be the number of unmatched applicants, we have an equal number of unmatched posts. In the second iteration each of these n_1 unmatched applicants of the first iteration picks his/her most preferred post independently and uniformly at random from the set of unmatched posts. This is identical to the first round except that we are now operating with n_1 applicants and n_1 posts. We repeat the same step as in the first iteration and continue this experiment till all applicants (and thus posts) are matched. Note that the number of iterations taken by this experiment to compute an $\mathcal{A}$-complete matching is an upper bound of the number of iterations taken by Algorithm Bounded-Unpop since here we arbitrarily assign an even applicant to an odd post, while in Algorithm Bounded-Unpop we make no such prior assignment. If this experiment finds an $\mathcal{A}$-complete matching in t iterations, then our algorithm certainly finds an $\mathcal{A}$-complete matching by t iterations, if not sooner.

In general, at the beginning of the i-th iteration, we have n_{i-1} applicants and n_{i-1} posts and each of these n_{i-1} applicants picks his/her most preferred post independently and uniformly at random from these n_{i-1} so far unmatched posts. Assuming that the post p_j is still unmatched at the beginning of the i-th iteration, we have:

$$Pr(p_j \text{ does not get matched in the } i\text{-th iteration}) = \left(1 - \frac{1}{n_{i-1}}\right)^{n_{i-1}} \leq \frac{1}{e}.$$

Using this, it is easy to see that:

$$Pr(p_j \text{ is unmatched in each of iterations } 1, \ldots, i) \leq \frac{1}{e^i}.$$

Thus the probability that p_j is unmatched in each of iterations $1, \ldots, \ln n_0$ is at most $1/n_0$. Hence the *expected* number of unmatched posts after $\ln n_0$ iterations is at most $\sum_{i=1}^{n_0} 1/n_0 = 1$. Hence the expected number of iterations required for our random experiment to terminate is bounded by $\ln n_0 + 1$. We can conclude the following theorem.

Theorem 4. *The expected number of iterations taken by Algorithm Bounded-Unpop to find an $\mathcal{A}$-complete matching in a random instance with n_0 applicants and n_0 posts, where each preference list is a uniform random permutation of the n_0 posts, is at most* $\ln n_0 + 1$.

6 Conclusions and open questions

We considered the problem of computing a matching in $G = (\mathcal{A} \cup \mathcal{P}, \mathcal{E})$ whose unpopularity can be bounded. We showed a sequence of graphs $H_2, \ldots, H_k, \ldots$ such that if H_k admits an $\mathcal{A}$-complete matching, then we have a matching M_k whose unpopularity factor is at most $k-1$. We implemented our algorithm and ran it on random instances, where we observed that we always found an $\mathcal{A}$-complete matching in H_k for $k \leq 4$. It is an open question to theoretically show that random instances admit low unpopularity factor matchings. We showed that the expected number of iterations taken by our algorithm on random instances with an equal number n_0 of applicants and posts is at most $\ln n_0 + 1$.

Acknowledgments. We thank the reviewers for their helpful reviews and comments.

References

1. A. Abdulkadiroğlu and T. Sönmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3):689–701, 1998.
2. D. J. Abraham, K. Cechlárová, D. F. Manlove, and K. Mehlhorn. Pareto-optimality in house allocation problems. In *Proceedings of 15th Annual International Symposium on Algorithms and Computation*, pages 3–15, 2004.
3. D. J. Abraham, R. W. Irving, T. Kavitha, and K. Mehlhorn. Popular matchings. *SIAM Journal on Computing*, 37(4):1030–1045, 2007.
4. D. Gale and L. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–14, 1962.
5. P. Gärdenfors. Match making: assignments based on bilateral preferences. *Behavioural Sciences*, 20:166–173, 1975.
6. R. L. Graham, M. Grotschel, and L. Lovasz. Chapter 3, matchings and extensions. In *The Handbook of Combinatorics*, pages 179–232, 1995.
7. D. Gusfield and R. W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, 1989.

8. A. Hylland and R. Zeckhauser. The efficient allocation of individuals to positions. *Journal of Political Economy*, 87(22):293–314, 1979.
9. R. W. Irving. Stable marriage and indifference. *Discrete Applied Mathematics*, 48:261–272, 1994.
10. R. W. Irving, T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch. Rank-maximal matchings. *ACM Transactions on Algorithms*, 2(4):602–610, 2006.
11. T. Kavitha, J. Mestre, and M. Nasre. Popular mixed matchings. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming*, pages 574–584, 2009.
12. T. Kavitha and M. Nasre. Note: Optimal popular matchings. *Discrete Applied Mathematics*, 157(14):3181–3186, 2009.
13. T. Kavitha and M. Nasre. Popular matchings with variable job capacities. In *Proceedings of 20th Annual International Symposium on Algorithms and Computation*, pages 423–433, 2009.
14. M. Mahdian. Random popular matchings. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, pages 238–242, 2006.
15. D. Manlove and C. Sng. Popular matchings in the capacitated house allocation problem. In *Proceedings of the 14th Annual European Symposium on Algorithms*, pages 492–503, 2006.
16. R. M. McCutchen. The least-unpopularity-factor and least-unpopularity-margin criteria for matching problems with one-sided preferences. In *Proceedings of the 15th Latin American Symposium on Theoretical Informatics*, pages 593–604, 2008.
17. E. McDermid and R. W. Irving. Popular matchings: Structure and algorithms. In *Proceedings of 15th Annual International Computing and Combinatorics Conference*, pages 506–515, 2009.
18. J. Mestre. Weighted popular matchings. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming*, pages 715–726, 2006.
19. A. E. Roth and A. Postlewaite. Weak versus strong domination in a market with indivisible goods. *Journal of Mathematical Economics*, 4:131–137, 1977.
20. Y. Yuan. Residence exchange wanted: a stable residence exchange problem. *European Journal of Operations Research*, 90:536–546, 1996.
21. L. Zhou. On a conjecture by gale about one-sided matching problems. *Journal of Economic Theory*, 52(1):123–135, 1990.